

Trabalho Prático 1

Hotel SweetDreams

Estudantes:

Bárbara Pinto 1191507

João Nogueira

Pedro Vieira

Grupo:

5

Turma:

2DL

Professor:

Nelson Freire

Disciplina:

Bases de Dados

# Introdução

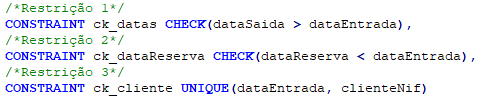
Neste trabalho prático aplicamos os conceitos de modelos relacionais, normalização e programação em Structured Query Language (SQL) para desenvolver uma base de dados para um hotel, contendo registos de reservas, clientes, quartos, funcionários, intervenções, etc.

# Modelo Relacional

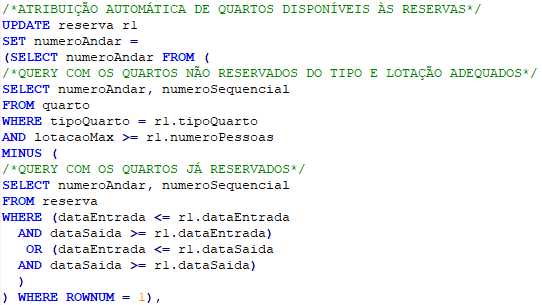
O modelo de dados relacional apresentado abaixo (*Figura 1*) apresenta a identificação de todas as entidades relacionadas com o problema colocado, juntamente com os respetivos atributos e relações que estabelecem entre si.

Como restrições de identidade que não podem ser identificadas no modelo, destacamos:

* Na tabela **Reserva**, a necessidade do *check-out do quarto ser após o check-in* - restrição 1 -, assim como da reserva ser efetuada com antecedência (antes do *check-in*) - restrição 2. Por último, cada cliente apenas pode ter um quarto reservado de cada vez - restrição 3.

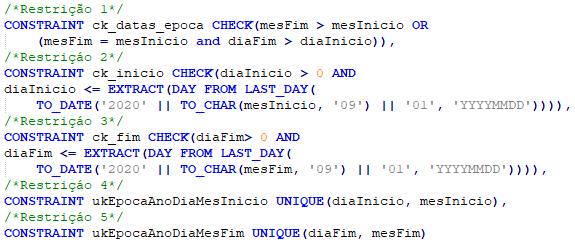


Adicionalmente, torna-se necessário garantir que não é possível reservar o mesmo quarto para a mesma altura a dois clientes diferentes. Neste caso, escolhemos não usar CONSTRAINTs no script de criação das tabelas. Em alternativa, definimos o número de quarto (número de andar + número sequencial) como NULL por defeito, o que representa a criação de uma reserva sem atribuição de quarto. Posteriormente, no script de inserção de linhas nas tabelas, fazemos o UPDATE das reservas, verificando quais os quartos disponíveis que cumprem os requisitos e atribuindo automaticamente.



*Exemplo parcial do código utilizado*

* Na tabela **EpocaAno**, as restrições dos meses e dias obrigam a que se encontrem compreendidos entre 1 e 12 meses, e entre 1 e 31 dias, respetivamente – restrição 2 e 3. Garante-se, ainda, que as datas das diferentes épocas do ano não se sobrepõem – restrição 6 e 7 – e que a altura de fim de cada época acontece após o seu começo – restrição 1.



* Na tabela **Manutencao**, a única restrição de integridade consiste em garantir que todas as combinações de quarto, funcionário e data são únicas, isto é, cada funcionário apenas pode estar num dado quarto num dado momento do dia.



Algumas considerações:

* Considerou-se que a lotação máxima de cada quarto é exclusiva, isto é, não depende do tipo de quarto. No enunciado é indicado “Os quartos têm uma lotação máxima, podem ser de diferentes tipos e pertencem a diferentes andares”, o que parece indicar que a lotação máximo é uma característica do quarto em si e, de acordo com a indicação do docente, escolhemos interpretar da maneira que achamos mais adequada. Podemos justificar esta aplicação no mundo real como a capacidade de instalar uma cama extra em certos quartos ou não.

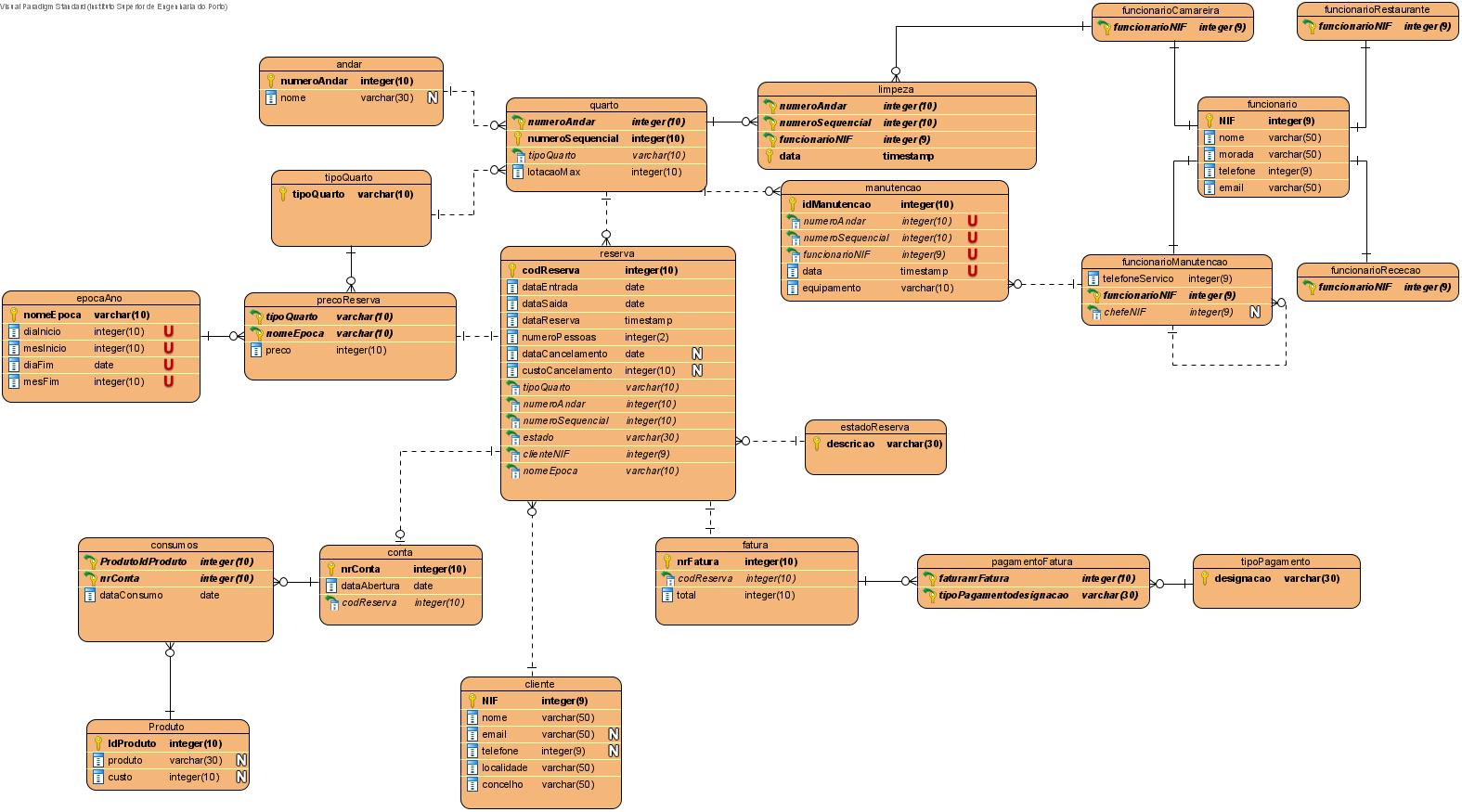


Figura 1: Modelo Relacional

# Consultas

Parte I

Alínea a)

Apresentar todos os pedidos de intervenção em aberto (intervenção ainda não registada) alocados a funcionários de manutenção que não fizeram nenhuma intervenção (registo) nas últimas 48 horas.

Neste caso, a solução desenvolvida envolveu apenas o uso da clausula de exclusão NOT IN e uma simples subquery para obter todos os funcionários identificados em manutenções com data compreendida nos últimos dois dias.

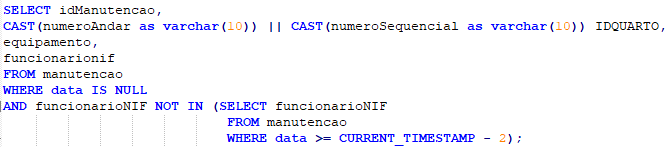


Figura 2: Código SQL usado para a alínea a).

Alínea b)

Apresentar a data, a hora e o nome dos clientes que reservaram quartos somente durante o mês de Abril e Junho deste ano. No caso de algum cliente ter reservado um quarto do tipo suite, deverá apresentar a localidade desse cliente numa coluna intitulada “Zona do País”.

Para a resolução, recorreu-se a uma UNION entre os registos de quartos do tipo suite e os restantes tipos de quarto para obter o resultado pretendido na coluna “Zona do País”, como mostrado abaixo.

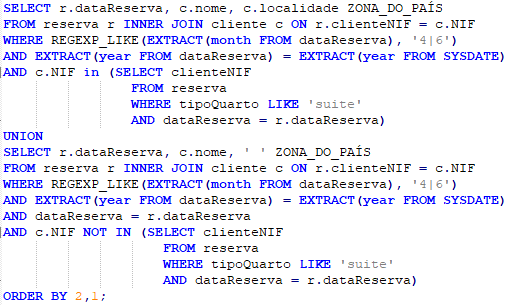


Figura 3: Código SQL usado para a alínea b).

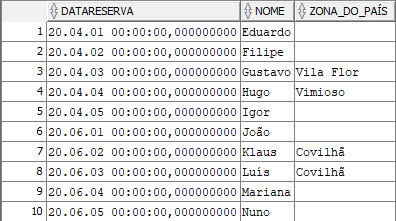
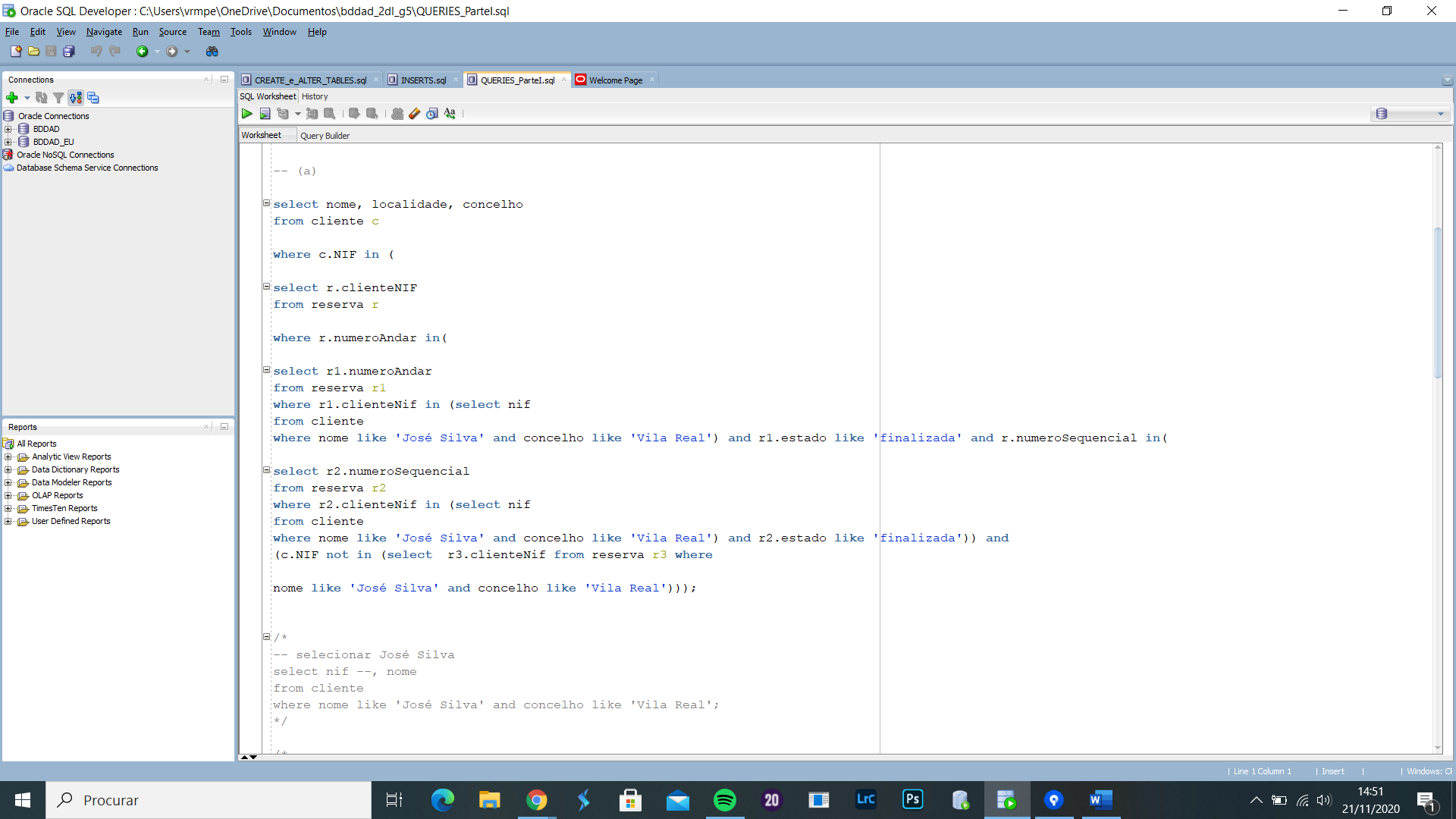


Figura 4: Resultado da alínea b).

Parte II

Alínea a)

Neste exercício, era pedido que apresentássemos o nome, a localidade e o concelho dos clientes que já estiveram alojados nos quartos já reservados pelo cliente cujo nome é José Silva, que é do concelho de Vila Real, considerando apenas as reservas “finalizadas” do cliente José Silva. Assim sendo, de forma a cumprir com o que era requisitado, apresentámos a seguinte solução:



Começamos por procurar quais os quartos já reservados pelo cliente José Silva, de Vila Real, selecionando o número do andar e o numero sequencial dos quartos que estiveram reservados por José Silva, sendo que a reserva já deve ter sido finalizada. Seguidamente, verificamos quais são os clientes que fizeram as reservas desses quartos, através do seu NIF, excluindo todos aqueles que se chamam José Silva e pertencem a Vila Real. Sendo que apresentamos o nome, a localidade e o concelho dos clientes que correspondem a esse NIF.

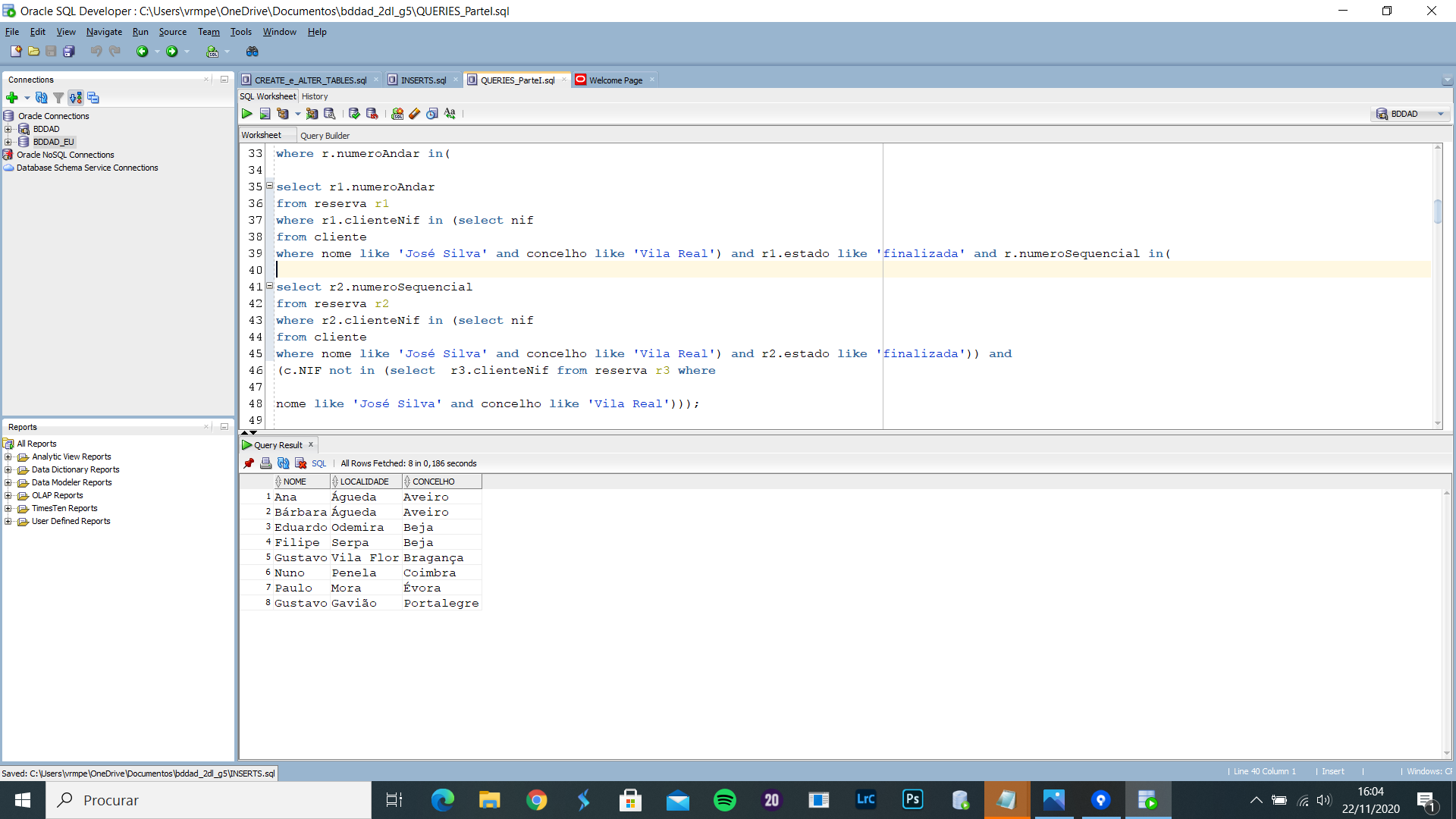
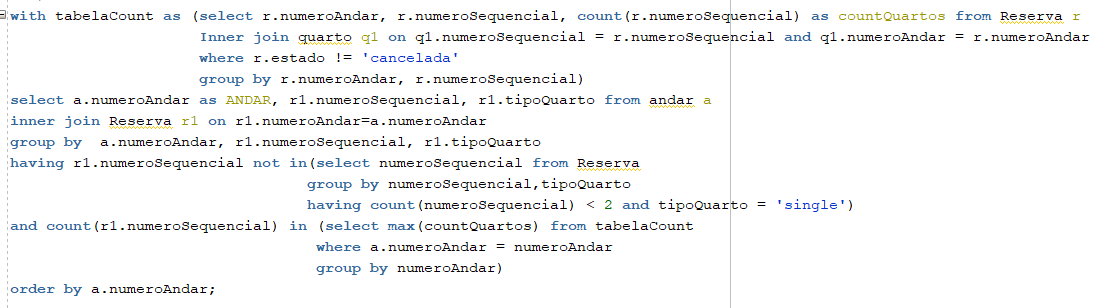


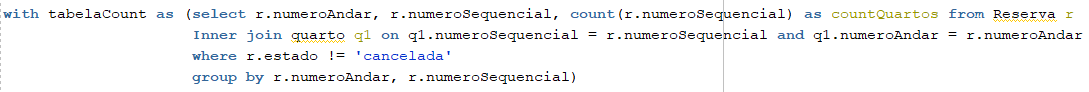
Figura 5 - Resultado da alínea a)

Parte III

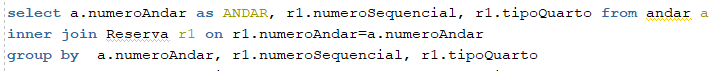
Alínea a)



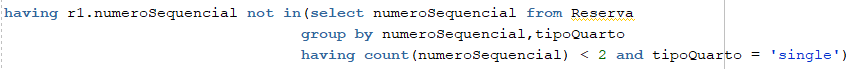
Criar uma tabela com o número de andar, o número sequencial e o número de vezes que o quarto foi reservado, em reservas que não foram canceladas:



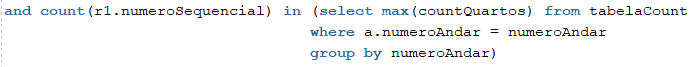
Selecionar o número de andar, o número sequencial e o tipo de quarto para quartos que foram reservados:



Remover os quartos que foram reservados, mas são do tipo “single” e foram reservados menos do que duas vezes:



Selecionar apenas os quartos que foram os mais reservados para cada andar:

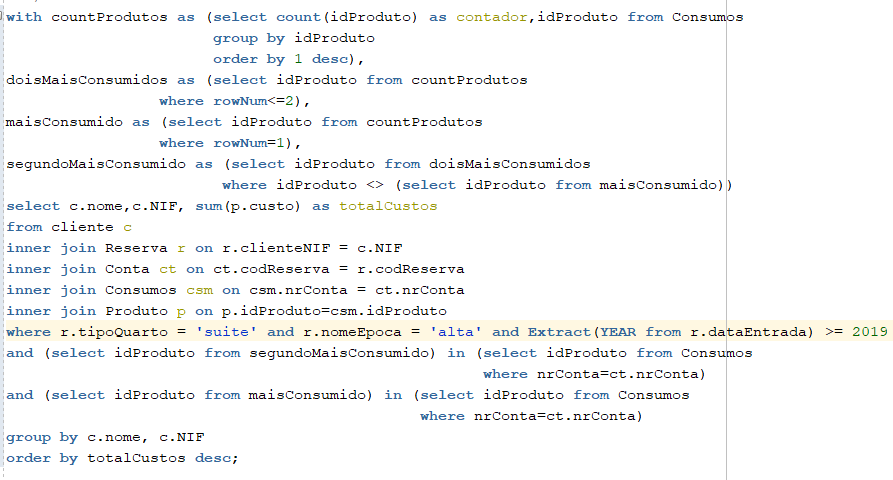


Ordenar o resultado por ordem crescente do número de andar:

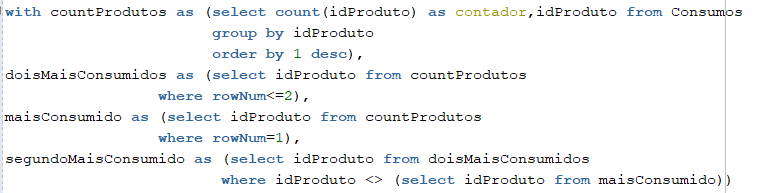


Resultado:

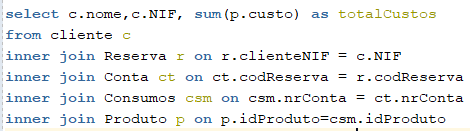
Alínea b)



Criar duas tabelas uma com o produto mais consumido e a outra com o segundo produto mais consumido:



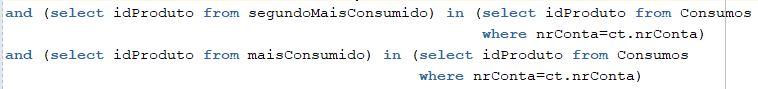
Selecionar o nome, o NIF e o total dos custos dos clientes:



Selecionar apenas clientes com reservas para quartos do tipo “suite” na época alta nos últimos dois anos:



Selecionar apenas clientes que consumiram os dois produtos mais consumidos:



Agrupar os resultados pelo cliente e ordenar pelo total dos custos:



Resultado:

# Conclusões

Ao longo deste trabalho deparamo-nos com algumas dificuldades, sendo que uma das principais passou por garantir a não sobreposição de reservas nos mesmos quartos para um mesmo período de datas.

A primeira abordagem com CONSTRAINTs mostrou ser insuficiente e, apesar da abordagem com UPDATE ter sido eficaz, foi necessário repetir o comando após cada INSERT, uma vez que a execução do UPDATE apenas no final fazia com que as linhas fossem reescritas com as mesmas atribuições de quarto, e não com quartos diferentes.

A solução a este problema poderia ser desenvolvida com PL/SQL, através de *triggers*, por exemplo, no entanto não temos ainda conhecimentos solidificados para a aplicar, tendo escolhido por isso algo menos eficiente, mas exequível.